

AN INTUITIVE USER INTERFACE FOR THE VIRTUAL REALITY RESPONSIVE WORKBENCH

Ranjeev Mittu
Naval Research Laboratory
4555 Overlook Avenue
Washington, DC 20375-5337

Justin McCune
ITT Systems and Sciences Corporation
2560 Huntington Avenue
Alexandria, VA 22303-1404

1. INTRODUCTION

The Naval Research Laboratory has been involved in research and development efforts in the areas of (1) automated routing of strike aircraft [1] (2) advanced 3D displays and interaction technologies, and (3) collaboration tools and protocols. This research has been conducted in order to advance the state of the art in each technology area, particularly for military command and control applications.

This paper will concentrate on the research conducted in item (2), the results of which have been incorporated in the STRike Optimized Mission Planning Module, STOMPM. The STOMPM testbed allows the user to load various terrain data, create strike related routing scenarios consisting of threats and targets, test the various routing algorithms and assess their performance, and visualize this information and interact with it in a natural setting. Two versions of STOMPM currently exist, version 1.0 (v1.0) and version 2.0 (v2.0). The STOMPM v1.0 was primarily developed to test auto-routing technology, and does not include technology needed to run on advanced displays such as the Virtual Reality Responsive WorkBench, VRRWB (i.e., STOMPM v1.0 operates best on a computer monitor with a mouse/menu interface). The STOMPM v2.0 also includes autorouting technology (specifically autorouting algorithms which take into consideration fuel and turn constraints which are not included in STOMPM v1.0), but also includes a user interface that is well suited to running on more advanced displays such as the workbench.

This paper will provide a high level description of both versions of the STOMPM testbed. Following this discussion, we will describe state-of-the-art technologies in visualization, advanced displays and scene interaction capabilities that have been incorporated within STOMPM v2.0. We will conclude with a discussion of the advantages associated with the use of the workbench and also the interface that has been developed within STOMPM for the workbench. Lastly, we will discuss areas for future research.

2. STOMPM SYSTEM DESCRIPTION

The STOMPM systems serves as a testbed for the research and development of strike asset routing algorithms, 3D displays and interaction techniques, and research in collaborative tools and protocols. Version 1.0 of STOMPM was built mainly to support the research being conducted in the area of strike asset routing algorithms, while version 2.0 of STOMPM was built primarily to support advanced 3D displays and interaction techniques (both versions have similar models, e.g., RTM [2]). The following sections will provide details associated with each version of STOMPM.

2.1 THE STOMPM v1.0

The original STOMPM testbed (V1.0) [3] was developed in C using the FORMS [4] software library for the user interface and the native SGI graphics library for rendering the scene. The main emphasis of STOMPM V1.0 is to allow the developer to easily incorporate auto-routing technology and be able to test the algorithms via a simple mouse/menu user interface. Many autorouting algorithms were implemented ranging from simple least cost path to jointly optimal routing [1]. A screenshot of the STOMPM system is shown in Figure 1. This version of STOMPM allows the user to load terrain maps (i.e., Digital Terrain Elevation Data or DTED), place assets, radar types, and targets on the terrain, specify routing parameters, and eventually choose a particular routing routine to find route(s) from the assets to the targets. The user has the ability to save/load scene files, view the environment from various locations, get/change information about entities in the scene, and modify certain attributes associated with the visualization of this information.

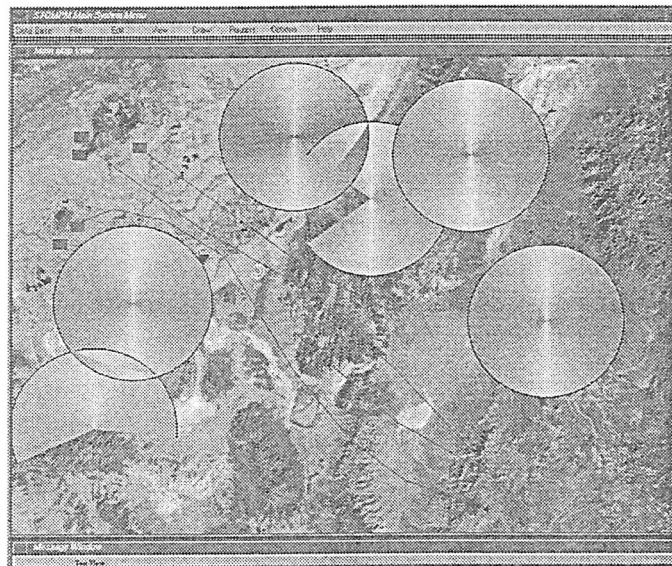


Figure 1: Screenshot of STOMPM V1.0 User interface. Shown are the RTM cones and a set of suppression and attack routes. The suppression routes open a corridor for the attack routes

Although version 1.0 of STOMPM was specifically designed to test autorouting technology, near the latter part of it's development cycle stereographics [5][6] were incorporated into the

interface. Since this version of STOMPM was still based on the mouse/menu style interface, it did not work well with stereographics because the menus were not drawn as a part of the environment being modeled, and only the environment and the objects contained in it were in stereo. This was part of the reason that STOMPM was eventually reimplemented. A second reason for reimplementation was due to the fact that version 1.0 was not easily extendable. Creating new objects meant defining new data structures for objects. An object oriented paradigm was investigated and eventually accepted as part of the design in order to provide a more flexible system in which new objects could be created with little effort, making use of already existing object classes.

2.2 THE STOMPM v2.0

The STOMPM V2.0 is a set of object-oriented C++ toolkits that facilitate the development of virtual environment simulation & planning environments. The STOMPM toolkits as they exist now provide a means to access and use routing algorithms, visualize and interact with a scene, and collaborate with other distributed STOMPM modules. A primary goal of STOMPM was that it be useful and easily transferable to other uses or designs. Towards this end, each of the STOMPM toolkits is specific in purpose, either extending existing functionality or adding to it. Through the combined use of some or all of the toolkits, it is possible to easily build new applications and/or interfaces by adding to the existing foundation of components and coding practices. In the discussion that follows, all italicized words can be interpreted as base classes or objects derived from those base classes. In either case, the meaning of these italicized words are the same in the context in which they are used, the difference is important only in the design and implementation phase (i.e., they can be used interchangeably for the purpose of the discussion).

The purpose of STOMPM is to support the capability to generate an automated set of routes from a set of starting points (sources) to a set of targets (sinks) contained in a defined scene with obstacles. A primary goal of STOMPM V2.0 is that it continue to be an extensible application, ready for use or as the basis of new or old applications. In achieving these two goals, as with all programming, it is important to define the constraints, key dependencies, and post-conditions that define how to implement the goals more concretely. Therefore, the design of STOMPM started with the desired capabilities, which were mainly set at the beginning, but also evolved over the course of the project. Below we discuss the capabilities developed as a result of our goals.

The routing algorithms in STOMPM, given a correctly specified scene and set of routing constraints, calculate an optimal route from a source to a target. Currently there are three algorithms in V2.0, but this number will be extended in the future. There is an unconstrained router, a router that is restrained by turn-angles but trades speed for a possibly non-optimal route, and an optimal turn-angle constrained router. The algorithms are interested in those objects in the scene that represent a threat to an asset attempting to get from the source to the sink. An example of such a threat is a radar, which interacts in complex ways with the

surrounding terrain. Thus the router is dependent upon a certain scenario or scene the user has created, and the underlying objects that make up this scene.

STOMPM provides several different utilities that may be used to create a versatile, alterable, and extendable user interface to interact with the objects in a scene or other components of STOMPM. The first point regarding the STOMPM interface is its ability to provide a representation of the scene which the router will use, implying both a user viewpoint, and graphic representations for all objects that make up the scene. Visualization is crucial for concepts like RTM that are most intuitively understood and correctable when visualized. Secondly, the ability to alter the user viewpoint and the objects in the scene (hereafter referred to as *SceneObjects*) is present in the form of a user *HotSpot*. The *HotSpot* is a 3D version of a mouse pointer and can be driven by a variety of input devices (mouse, 6 degree of freedom tracker, keyboard, etc.) and is used to select, move objects, alter the user's view, etc. STOMPM also is able to store and retrieve scenes for use at a later time.

Though the interactions with the *HotSpot* can be varied in several ways, there is still a need for other forms of input. STOMPM also provides both keyboard support and 3D menus (which interact with the *HotSpot*). All of this assumes that the user wishes to interact directly with the scene. However, another means of changing the scene the router uses is also available within STOMPM. There exists support for distributed communication, currently limiting the users to one concurrent shared scene. Thus, it is possible to alter the layout of the scene by reading from a STOMPM feed coming from another computer over a network. The interactions with the scene are merely support to supply the router with the necessary information to do its work.

The STOMPM system is composed of a complex set of components. It is able to provide scene management, viewing, support for various input devices, a 3D menuing system (Figure 2), peer-to-peer networking support, object interaction through the *HotSpot*, provide information feedback, archivability, as well as providing hooks for other useful operations. The next section will describe the visualization, advanced display, and scene interaction capabilities that have been implemented within STOMPM v2.0.

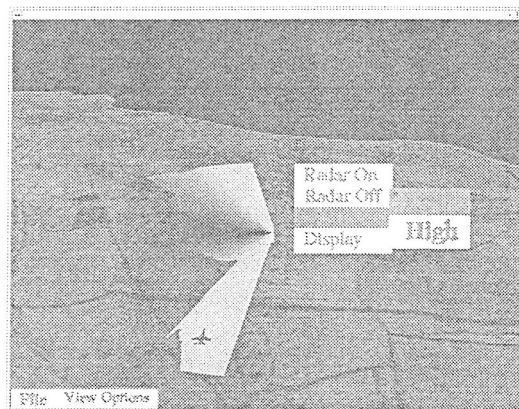


Figure 2: The STOMPM v2.0 user interface showing the *HotSpot* (seen as circle in "Low" per object submenu), 3D System Menu, and 3D Per/Object menus for manipulating object specific information

3. VISUALIZATION, ADVANCED DISPLAYS AND INTERACTION IN STOMPM

The next section will discuss a visualization technology that has been investigated and implemented within STOMPM, namely the use of stereographics for the inspection of the 3D strike information. The following section will discuss an advanced display technology being utilized for the 3D visualization of STOMPM scenarios, namely the virtual reality responsive workbench. The advantages of using the workbench will be presented. Lastly, we will discuss novel interaction technologies that have been developed within STOMPM for the workbench.

3.1 VISUALIZATION TECHNIQUES

Stereographics [5][6] provides a true 3D representation of an environment by providing two images to the eyes in sequential order, one for the left and one for the right. This allows the user to perceive depth on a two dimensional monitor. Implementing the stereo effect in software is not very difficult and works as follows: produce two images of the scene and double the monitor's refresh rate. One of these images is for the right eye while the other is for the left eye. The user can then wear Liquid Crystal Display (LCD) shutter glasses to view the image in stereo. The shutter glasses work by showing the left eye the image intended for the left eye and the right eye the image intended for the right eye, in alternating sequence (i.e., the shutters in the glasses open and close in synchronization with the monitor's refresh rate - the synchronization signal is sent to the glasses from an emitter). The overall effect to the user is a view which more closely resembles 3D - the stereo image can be either projected in front of, or behind, the computer screen, by adjusting a parameter in the software that controls the distance from the eyes to the image convergence point.

There are several items worth mentioning about the use of stereo. As was already mentioned, the stereo image can be projected in front of, or behind, the computer screen by adjusting a certain parameter in the software. When one sets the parameter such that the image is projected in front of the screen, the eyes can get confused by a floating image in front of the screen, which when seen in comparison to the edges of the window/display, appear underneath the window/display (edge effects). Zooming or panning effects can further magnify the "edge effect" phenomenon. Therefore, it is important to have the entire scene visible when one wishes to project the image in front of the computer screen. However, by having the entire scene visible on the screen, it may be impossible to view the important details associated with the scenario. In many instances, it is desirable to project the image behind the screen. What we have noticed is that in our particular application, when the maximum height of the viewable terrain is projected behind the screen, even after panning or zooming, edge effects are removed. In this regard, it may be easier to zoom or pan, thus enabling more details associated with the scenario to be seen.

3.2 ADVANCED DISPLAYS

The NRL has investigated the use of advanced displays for Command and Control applications, particularly the use of a virtual reality responsive workbench, (Figures 3a and 3b). The workbench was originally developed and built at the GMD National Research Center for Information Technology, and a copy was built at NRL for initial research. Currently, NRL is using a commercially available workbench developed by Fakespace Corporation. Whereas the original workbench top was not adjustable, the one developed by Fakespace has an adjustable table top which can tilt to approximately 45 degrees for easier viewing.

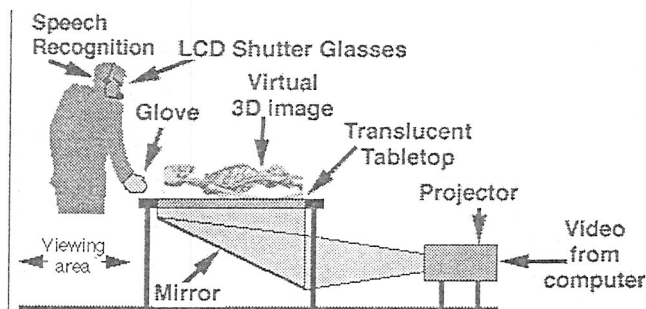


Figure 3a: A schematic of the virtual reality responsive workbench (printed from the NRL's VR Laboratory homepage)

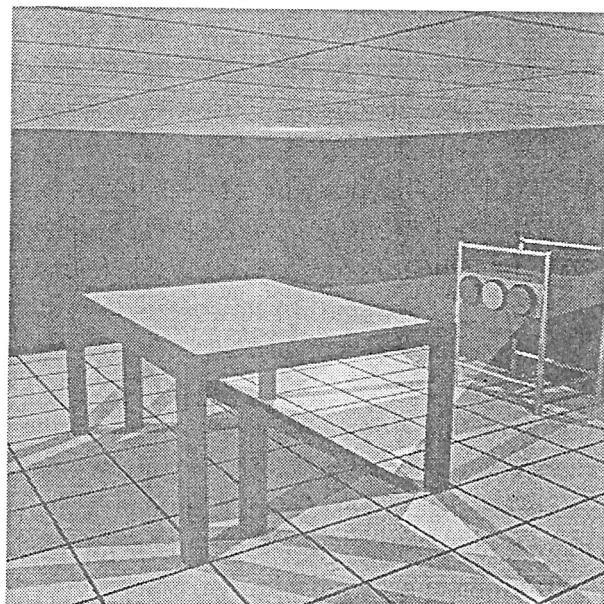


Figure 3b: A computer generated image of the virtual reality responsive workbench

Video from the computer is sent to the projector, which projects the stereo image onto a mirror. The mirror reflects this image onto a translucent table top. Two pairs of emitters are mounted at the back two corners of the table top. The use of two pairs of emitters provides a stronger synchronization signal for the shutter glasses, however, a single emitter could be configured for use. When displaying an image on the workbench, the user can control whether the image appears to float above, or just below, the table top by adjusting the same parameter which controls where the image converges (e.g., with respect to the far/near clipping planes).

The advantage of the use of stereo on the virtual workbench arises from observing that users naturally perceive altitude in the same direction as a vector which is perpendicular to the earth. The workbench provides an environment in which users can naturally interact with objects on terrain in a natural table top environment. On a computer monitor, altitude would be in the same direction as a vector perpendicular to the computer screen. This is a little awkward to work with, especially when trying to adjust the routes in the z direction. The viewing area is also

greater as compared to the standard computer monitor. This larger viewing area provides a more comfortable environment for multiple user to interact with the workbench application.

3.3 INTERACTION IN STOMPM v2.0

STOMPM is made up of several components and toolkits. Each toolkit/component provides an independent functionality that is intrinsically different from the others. The primary toolkits that STOMPM provides are the Application Tool Kit (ATK), the Routing Tool Kit (RTK), the Graphic ATK (GATK), and the Networking TK. Within each of these toolkits are components and/or extensions to objects in other toolkits. Another crucial component is the *SceneObject* that the other toolkits either manage or use to accomplish their task.

The ATK provides a framework basis for the other toolkits. The GATK adds graphic capabilities to the existing system. The RTK uses the ATK and *SceneObjects* to perform its routing tasks. The Networking TK extends network functionality to the ATK & GATK, but could also be considered more an extension of the existing framework than a part of the framework. The GATK will now be described in greater detail.

The purpose of the GATK is to provide a base set of tools that are extendable for developing the GUI that will meet the end-users' needs. The GATK, as its name implies, inherits from the ATK, mainly because all of the components it uses must also be changed to be graphic in nature but are the same in functionality. A platform limitation of the current implementation is that the GATK uses Performer, an SGI rendering system, to do its rendering and much of its interfacing. Beyond these factors, the GATK extends the ATK to include input devices, viewing paradigms, interface mechanisms, and other IO components which are usable in part, whole, or not at all, based on user requirements.

The key extension of the GATK is the introduction of the *HotSpot*. The *HotSpot* corresponds to a 3D mouse pointer in concept. With the *HotSpot*, you can select, pop menus up, move objects, etc. It is the graphical means by which input is specified. Visually it is represented as a spot on the screen, which is simple enough to understand. The more difficult part, is how to move and position it with 6 degrees of freedom, and how it communicates with the objects it interacts with.

The *HotSpot* currently is driven by one of two input devices. The first input device is a 3-Button mouse, and the second input device is a 6 degree of freedom tracker. The tracker is currently used as a virtual pointing stick to obtain the point of interest on the screen. The point of interest for a mouse is likewise an x-y coordinate pair. The point of interest is used to project a ray from the users viewpoint to the viewing screen's position in world-space, and the first object of intersection (assuming intersection) becomes the location of the *HotSpot*. So in a very real sense, the *HotSpot* is its own device that is driven by the tracker or the mouse. Using the buttons, the tracker or mouse handler can move the *HotSpot* in or out. As the *HotSpot* is

considered native to the GATK, whereas the tracker/mouse devices are not, the *HotSpot* has communication protocols established with the objects it interacts with.

Every object of significance on the display is given the ability to handle events. By this mechanism, all *GSceneObjects* (and *InterfaceObjects* such as menus) are given the ability to respond to events that relate to them. This is set up primarily for interactions with the *HotSpot*, though the developer has the option of extending this. The *HotSpot* thus at agreed times sends informational messages to the objects it interacts with. In particular, when the object is selected, unselected, hit, or unhit, the object in question is notified that the event took place. Thus, each object chooses how it will respond to particular events. Most objects will probably respond in the same way, and thus are given a default handler. For instance, when most *SceneObjects* receive a move event, they move themselves in space. However, when the *TerrainObject* receives a move event, its behavior is overridden to move the viewer of the scene thereby accomplishing the desired interaction. Thus every object has the chance to easily override behavior.

By using the *HotSpot* and its communication protocol, several features of interest have been added to STOMPM. The user can now add 3D menus that when selected will perform developer specified callbacks. These menus are also operable on a per-object basis, radars can have one type of menu, each tank can have its own specialized menu, etc. Since these menus are just like any other 3D object in the scene, the stereo effect is preserved. Objects can have designated common handlers, e.g. all objects that must be placed on the ground can use one common handler that drops the object back to the ground when left in the air, while objects that can be left in the air use a different handler. An important aspect of a good user interface is the ability to move through the displayed scene quickly and easily.

STOMPM currently has two chief modes of viewing the scene. These modes are tethered viewing and egocentric viewing, which are entered into by selecting the menu items under the viewing menu. In tethered viewing mode, the user is virtually tethered to the 3D point of interest in world space, which is where the *HotSpot* was before entering viewing mode. In this case, moving to the left entails rotating about the 3D point of interest at a fixed radius in the XY plane that corresponds to the viewer's left. Viewing in egocentric mode is simpler. When the camera "moves" to the left, the viewer rotates their view and does not translate at all. Tethered viewing mode is of primary use when examining an object from several angles, but always looking towards the same area. Egocentric viewing mode is always looking from the same area. A third useful application of egocentric viewing is the ability to jump to any object the *HotSpot* points to, and then view from that point. Thus, it is possible to jump to a pilot's view or a view from a certain hilltop.

By using these two interface mechanisms, and other aspects of the GATK and *GSceneObject* interface it is possible to quickly maneuver through a scene and view its objects, as well as relocate them. Currently missing from the framework is an informational feedback of where the objects are as they are being moved, though this could be implemented quickly enough. Other

features, such as drop-lines from the objects to give the user positioning information when they float above the terrain, bins to temporarily place objects in, means of removing all objects of a given type, etc. are all possibilities with the current STOMPM. However, as may have been apparent from the discussion there are some features that are more a part of the toolkit than others, and it is questionable where to draw the line between the toolkit components and objects proper, and those specific to the application. The end choice is always up to the developer.

4. CONCLUSION

The visualization of strike related information such as aircraft routes, terrain and radar envelopes has been enhanced by the use of stereographics, which has made it easier to view depth. Furthermore, coupling stereographics with an advanced display technology such as the workbench allow the users to work with a true 3D representation of the world on a table top environment, which appears to be most natural for planners as many of the planning activities with maps, etc are done on table tops. The workbench also provides a greater viewing area, allowing multiple participants to view and potentially interact with the environment being modeled. The interface that we've developed within STOMPM works well with the virtual workbench in many respects. Because the menu system is 3D, it does not interfere with the stereoscopic workbench application as would the traditional menus. Secondly, the object reachability constraints make it more convenient to drive the *HotSpot* via a joystick as opposed to driving it via other devices on the workbench such as data gloves. Due to the large display area associated with the workbench, a person wearing a data glove may find it difficult to interact with objects placed over such a wide display area. Using the joystick and associated buttons, the user is able to rapidly point and click with the *HotSpot* anywhere on the display area.

5. FUTURE DIRECTION

The GATK has a wealth of opportunities for expansion. It currently stores none of its parameters to file, so it is not customizable at all. There are many concerns regarding the display of other information, and options that are possible in the VR world that go beyond the components common to the X environment and personal computer operating environments. It would also be interesting to provide X widgets and menus, and/or porting to non-SGI systems, which would require a hefty rewrite of all the graphical elements. A potential area for investigation is the use of web technologies such as Java/VRML as a graphical front end for STOMPM. This would allow planners to interactively plan and collaborate with workbench planners. We have replicated a module within STOMPM using Java and plan to continue research in the use of this emerging technology for real applications. Another area that we're investigating is the potential for the use of immersive environments in conjunction with the workbench. A hypothesis is that planning functions would work best in an environment which allows "Gods Eye" viewing such as that provided by the workbench. Also, once generated, these plans would best be simulated in immersive environments. We are actively looking into issues pertaining to the interface between these two different display platforms. We are also investigating multi-modal interfaces to replace or complement the already existing interface within

STOMPM. A multimodal interface will be of particular importance in immersive environments in which there may be a varied and large amount of data to navigate through.

6. REFERENCES

- [1] Zuniga, M., "The Interdependent Joint Routing Problem: Description and Algorithmic Approach", *Proceedings of the Eleventh Annual Command and Control Decision Aids*, Monterey, CA. 1994.
- [2] Mittu, R., Uhlmann, J.K., "Strike Visualization in Stereo on the Virtual Workbench", *Proceedings of the Sixth Annual Workshop on Information Technology and Systems*, Cleveland, OH., 1996.
- [3] Mittu, Ranjeev., "A Prototype System for the Evaluation of Interdependent Routing Algorithms for Military Aircraft", *Proceedings of the 11th Annual Decision Aids Conference*, Naval Postgraduate School, Monterey, CA., *Precision Strike Technology Symposium*, Johns Hopkins University Applied Physics Laboratory, Columbia, MD, 1994.
- [4] Overmars, Mark H., "Forms Library: A Graphical User Interface Toolkit for Silicon Graphics Workstations", Department of Computer Science, Utrecht University, Netherlands., 1991.
- [5] Tessman, Thant., "Perspectives on Stereo", IRIS Universe, *Summer 1989*.
- [6] Akka, Robert., "Writing Stereoscopic 3D Graphics Software using Silicon Graphics GL", *StereoGraphics Corporation Internal report*, April 17, 1991.